A Software Engineering Paradigm as a basis for Enterprise Integration in (Multi-) Client/Server Environments

Dr. D. Solte
Research Institute for Applied Knowledge Processing (FAW) Ulm
Helmholtzstraße 16, 89081 Ulm, Germany, Tel: 0731/501-510,
Fax: 0731/501-111, solte@faw.uni-ulm.de

Abstract

The ability to build and execute enterprise models including data, service and process models is a topic of growing importance for industry. It addresses the problem to develop reasonable models of the enterprise but has to cope also broadly with implementation and execution issues in heterogeneous environments. With respect to implementation and execution, client/server architectures, request broker mechanisms and distributed data and applications are emerging as the future state-of-the-art. In this context, the existing heterogeneity of technological frameworks as well as coping with legacy systems is a crucial fact. Existing methodologies and tools are not overwhelming these problems. They often do not integrate aspects of enterprise- or process-modelling, CASE (Computer Aided Software Engineering), workflow management and client/server execution. As a consequence, new kinds of architectures are needed. This paper outlines an approach, developed at FAW for the described scenario. The main objective of this solution is to cope with heterogeneity by a neutralizing approach instead of standardization. The described software engineering paradigm supports a model-oriented development of distributed data, services and processes in a uniform way towards a neutralizing execution environment. The FAW software engineering paradigm complies with the specifications of CIMOSA and accomplishes the requirements of the CORBA architecture.

Keywords

Client/Server, Enterprise Engineering, Enterprise Integration, Execution Environment, Integrating Infrastructure, Operational Paradigm, Process Modelling, Software Engineering.

1 INTRODUCTION

Since ´87, the FAW (Research Institute for Applied Knowledge Processing, Ulm) has devoted continous efforts to develop a comprehensive framework for enterprise engineering and application development, which consists of an overall systems architecture and a pertaining software engineering paradigm (Holocher et al., 1993, Radermacher and Solte, 1994). The most important aspect when developing this overall strategy is the observation that none of the relevant standardization activities will really lead to a worldwide homogeneous standard for a wide body of services (Verall, 1991). As a matter of fact, heterogeneity will always prevail with respect to infrastructural components, but also in methodologies and other aspects of IT-technologies. Taking this into account, the main aim of the FAW software engineering paradigm is to define and implement a framework that actively supports enterprises in building an integrated and cooperative engineering and execution environment for distributed data, services and (business) processes. Related to the requirements described by (ESPRIT Consortium AMICE, 1993) (cf. Figure 1) this leads to the development of an integrating infrastructure named AMBAS (Adaptive Method Base Shell, c.f. Holocher and Solte, 1992) and an engineering environment named Ωmega (Operational Modelling Environment and Generator for AMBAS Applications).

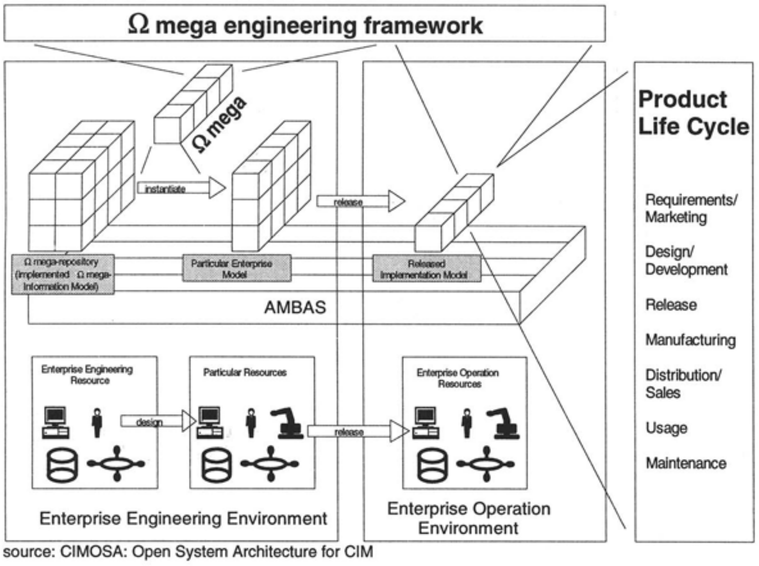
Ω mega engineering framework
Ω mega
instantiate
release
Ω mega-repository (implemented Ω mega-Information Model)
Particular Enterprise Model
Released Implementation Model
AMBAS
Enterprise Engineering Resource
Particular Resources
Enterprise Operation Resources
design
release
Enterprise Engineering Environment
Enterprise Operation Environment
Product Life Cycle
Requirements/ Marketing
Design/ Development
Release
Manufacturing
Distribution/ Sales
Usage
Maintenance
source: CIMOSA: Open System Architecture for CIM

Figure 1 Positioning of Ωmega/AMBAS within CIMOSA.

An important issue for the design of the systems architecture was the consideration of migration needs of enterprises, since taking into account the existing infrastructural components and methodologies already used. This has lead to a so-called active federation strategy as a framework for a knowledge-based approach that tries to cover heterogeneity by neutralization instead of standardization. Neutralization is thereby performed by using models of the different components and translating the models using knowledge-based mechanisms as proposed in (Petrie (ed.), 1992).

Besides that more principal design aspect, the framework (c.f. Figure 2) consists of several parts which are combined in an overall architecture. The main parts are the information model, the process model (including methodological concepts) and the tool model. The tool model consists mainly of the repository, I/O editors, generators (Ωmega) and an execution environment (AMBAS).

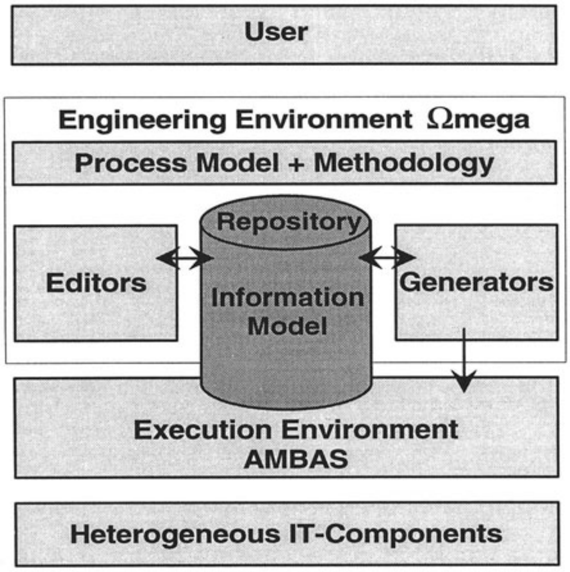
User
Engineering Environment Ωmega
Process Model + Methodology
Repository
Editors
Information Model
Generators
Execution Environment AMBAS
Heterogeneous IT-Components

Figure 2 Enterprise engineering and execution framework.

The FAW software engineering paradigm has its roots in project TIMM (Technical Infrastructure and MultiMedia). TIMM established the first heterogeneous and distributed computing environment at FAW (c.f. Figure 3), ranging from personal systems over workstations to a mainframe (Solte and Heerklotz, 1991). Moreover, these systems were connected to special equipment such as PABX and state-of-the-art multimedia components by using several networks (ISDN, Ethernet, Token-Ring, Broadband, Sinec-H1).

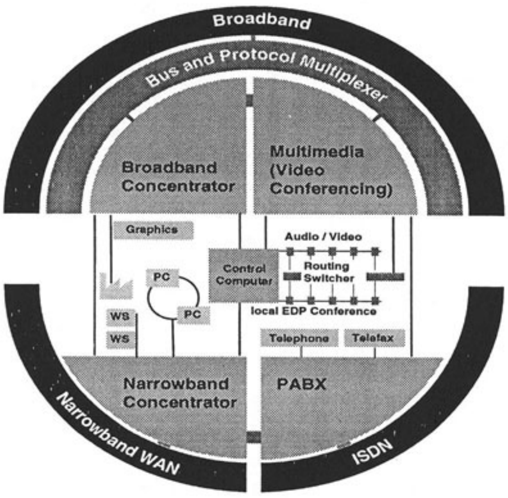

Figure 3 Heterogeneity at FAW.

The chief objective of the FAW software engineering paradigm is the definition of concepts and tool designs to support the cooperative development of distributed data and applications (including process-engineering and control) for this kind of heterogeneous platforms in particular consiering reliability aspects. The basic foundation has been laid by project MIDA (Model-Oriented Integration of Data and Algorithms) (Holocher et al., 1993) which continued the work of project SESAM (Decision Support for Job-Shop Scheduling) (Müller and Solte, 1994). Project KIWI 2000 (Communications Infrastructure 2000) further expanded the framework with special focus on the execution environment by incorporating several telecommunication devices and services. KIWI 2000, which has been finished in 1995, was the largest joint project of the State of Baden-Württemberg. It involved 14 industrial partners, 2 chambers of commerce and the German Telecom (Kopaczyk et al., 1993).

2 THE ENGINEERING PARADIGM

The FAW software engineering paradigm focuses on the development of distributed data, services and processes in (multi-)client/server architecture. By (multi-)client/server architecture we mean the possibility to distribute presentation, process logic and data management within heterogeneous environments, where these components are implemented as servers. One server can be executed by several clients, one client can have access to several servers and servers can act as clients.

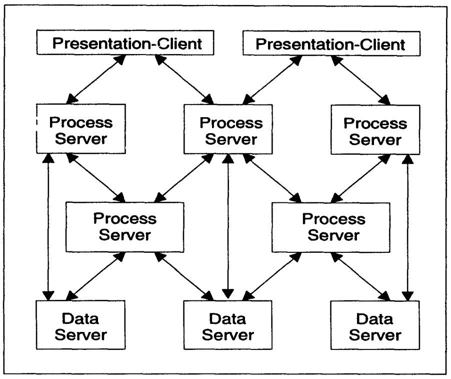

Figure 4 (Multi-)client/server architecture.

It is one important aspect that the binding of clients and servers should be possible during runtime, instead of compilation time. To support this kind of architecture, the execution environment contains mechanisms of a request broker (c.f. OMG, 1991) and the engineering environment consequently devides the applications logic from its implementation details.

Distributed data and methods (services, applications and processes) have to be modelled at an abstract model layer and will be translated automatically to corresponding (C++-) code after customizing the models due to implementation details. The compiled code can be executed by the neutralizing platform (AMBAS). In addition, one has to consider that information is needed to administrate productive implementations, and for systems management. To cover all these aspects, the information model comprises six representation layers, as depicted in Figure 5.

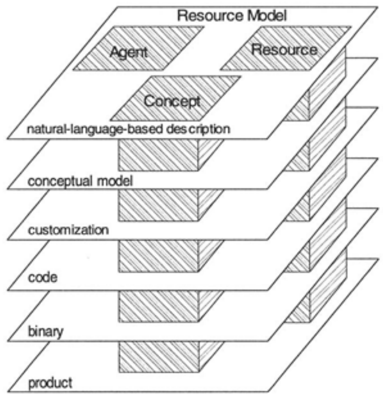

Figure 5 The Ωmega Information Model.

The first layer administrates natural language based descriptions. These are informal descriptions of all data and method components (services, applications and processes) of the enterprise (i.e. the ontology). The second layer represents the formal descriptions (models) of these components. At this layer - called the conceptual model - only the logical aspects of data, services, applications and processes are modelled. For reliability, reuse etc. it is important that no information about implementation details should be merged with the components logic. Instead, this information is modelled at a separate representation layer (customization). This includes technical and organizational aspects (communications infrastructure knowledge, e.g., GUI, CUI, file system, database management system, operating system, user model etc.) and incorporates the principle of definition moduls (conceptual model) and implementation models (customization) derived from modular programming concepts (c.f. Teufel, 1991) applied to models to the FAW software engineering paradigm. Using a conceptual model and a corresponding customization, a C++-representation is generated automatically and stored in the code-layer. The binary layer administrates the compiled components. The product layer captures information about installed (productive) data, services, applications and processes.

We have developed a specific strategy (the operational paradigm firstly introduced by Solte, 1987) to support the representation-framework (metameta-model) of this information model. We have chosen an approach with a modelling layer (analysis), with an orientation to rather classical concepts (data and functions) and with an implementation of data and functions in a strictly defined way as objects. Coming from a mathematical view functions are classified using their input and output signature leading to so-called function classes

$T(Y_i, Y_O) := \{T | T: Y_i \rightarrow Y_o\}$

where each concrete function represents an object of this class. Y_i specifies the functions input and Y_o the functions output. When implementing the function as an object, the function is encapsulated by a well-defined set of elementary access functions.

These are

create:	the creation of an object.
delete:	to delete an object.
assign:	the assignment of a value to the input of the function.
evaluate:	to activate asynchronously the evaluation of the function.
access:	to read the output of the function. If it is not yet evaluated, access also activates the method evaluate, otherwise access synchronizes the requesting process by waiting for the function to be terminated.
link:	to make the object only a reference to an already existing object.

When implementing data, we interpret them as special function classes $Y := \{id(y)$, e.g. $id: Y \rightarrow Y\}$. With this interpretation the same capsule, as it is defined for functions, can be implemented for data which leads to a uniform implementation and administration of data and functions for execution.

The operational paradigm overcomes problems using object-oriented principles "naively" for analysis (Holocher et al., 1994) but uses the full potential of object orientation for the implementation and execution of distributed applications. At FAW, we have developed a modelling language called OML (Operational Modelling Language) dedicated to support a reasonable combination of object orientation, semantic data modelling, a strict functional characteristic of the language (especially to support process modelling) and the capability to model constraints (but with a restricted constraint calculus). OML should mainly be perceived as the repository's metameta-model, it is not intended to be a solitaire modelling language for software engineering. We have already proved, that one could use different editors (supporting different languages) to build and modify models, e.g. STEP/Express (c.f. Anderl, 1993 and Grabowski, 1993) or GRAPES (c.f. Kaufmann, 1993).

The representation framework defined by the metameta-model OML allows the uniform knowledge representation of application concepts under different perspectives (Figure 6).

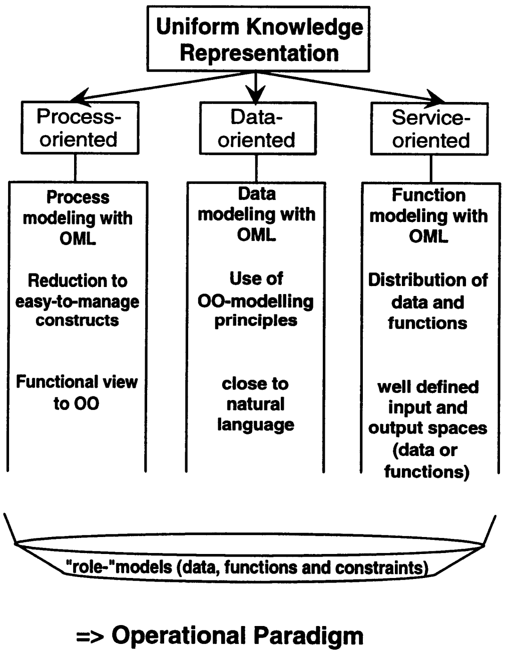
Uniform Knowledge Representation
Process-oriented
Data-oriented
Service-oriented
Process modeling with OML
Reduction to easy-to-manage constructs
Functional view to OO
Data modeling with OML
Use of OO-modelling principles
close to natural language
Function modeling with OML
Distribution of data and functions
well defined input and output spaces (data or functions)
"role-"models (data, functions and constraints)
=> Operational Paradigm

Figure 6 Uniform knowledge representation.

The strict functional characteristic allows the analysis and the requirements engineering in a process driven way but also on the basis of services (e.g., if specific technical servers like fax, telephone or others have to be modelled). On the other hand, the capabilities of semantic data modelling and object orientation allow the comfortable description of any kind of data. In addition, there is a possibility to collect data, functions and constraints in a context-oriented way by forming models, containing all these data, functions and constraints that fit to the specified context. This enables the integrated view of the different perspectives followed during modelling.

Part of the FAW software engineering paradigm is an analysis and design technique that forces the development of a model hierarchy, thereby allowing the different perspectives (e.g., process, service or data-oriented). At the bottom level, the models have to describe organizational or technical roles (by means of data that has to be provided; and functions the role is responsible for) of the enterprise. This leads to a role-based structure of the enterprise model (as depicted in Figure 7). A distribution logic could directly be derived from this structure. Role models are implemented as a whole and could be installed on all computers accessable by those organizations and agents that are capable of taking these roles. The structure within the information model, that supports the management of role-based enterprise models, is called resource model.

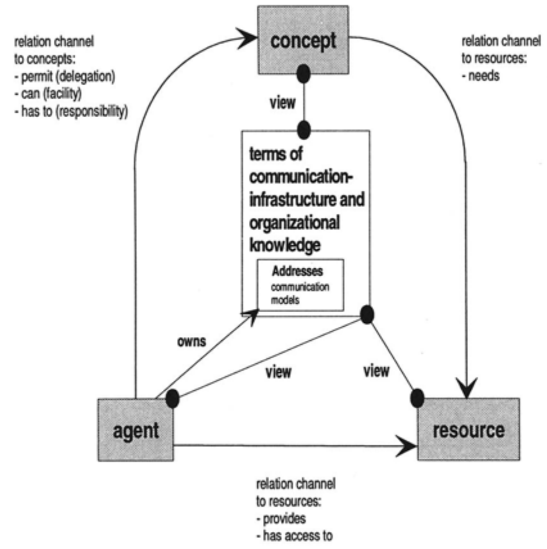

Figure 7 Construction of the Resource Model (Coarse Structure).

In the resource model all models are specializations of the term "concept" which has relations to terms "agent" and "resource". Agents are all acting objects (things that could take a role) of an enterprise which means technical systems like computers, printers, terminals and so on, but also humans or organizations. In addition, "concepts" need "resources" and an "agent" provides "resources". It is important that, based on these three top level terms of the enterprise´s ontology, one can define specific views forming the basis to model all kind of communication´s infrastructure and organizational knowledge. This is for example the definition of addresses as a possibility to describe communication models.

3 THE SYSTEMS ARCHITECTURE

The first operational prototype of the entire tool model for the FAW software engineering paradigm is now available. Ωmega (Operational Modelling Environment and Generator for AMBAS applications) contains specific components (in particular the repository/information model, I/O-Editors, generators and the neutralizing execution environment AMBAS) to develop models and to produce executable code. Available commercial products are integrated in Ωmega whereever appropriate.

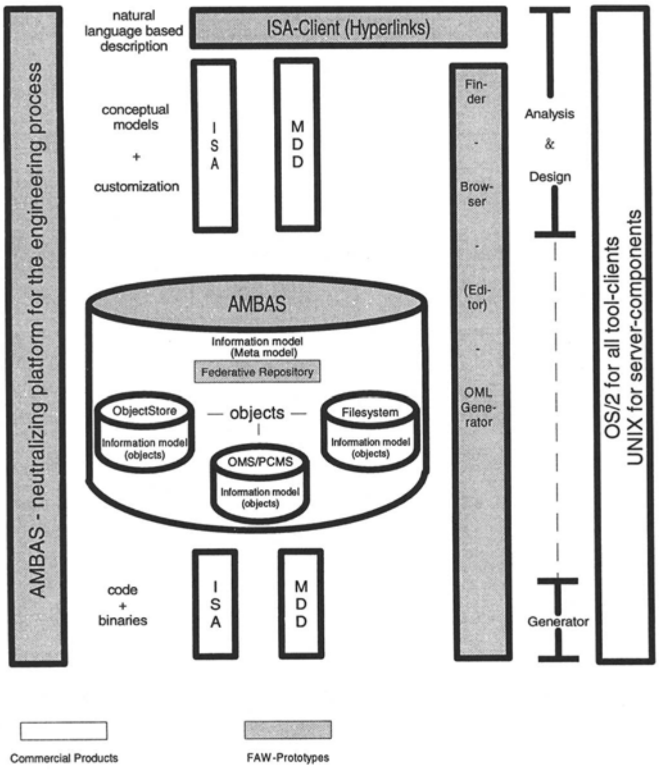
AMBAS - neutralizing platform for the engineering process
natural language based description
ISA-Client (Hyperlinks)
conceptual models + customization
ISA
MDD
Fin-der - Brow-ser - (Edi-tor) - OML Gene-rator
Analysis & Design
Generator
OS/2 for all tool-clients
UNIX for server-components
AMBAS
Information model (Meta model)
Federative Repository
ObjectStore
Information model (objects)
— objects —
Filesystem
Information model (objects)
OMS/PCMS
Information model (objects)
code + binaries
ISA
MDD
Commercial Products
FAW-Prototypes

Figure 8 Components of Ωmega.

As indicated in Figure 8, Ωmega provides different components to edit the different representation layers of the knowledge representation framework defined by Ωmega´s information model. For the natural language based description, we use a generic editor that we build for any kind of AMBAS-objects. Part of this editor is a generator that maps the structure of implemented objects onto a page-layout. Besides the implementation of this editor with Ωmega (which means the use of the ISA dialogue manager for the GUI) we have developed a low-cost variant based on HTML. This allows to edit AMBAS-objects with any kind of HTML-editor (e.g. MOSAIC). Applying this to the natural language based description every page generated relates to an ontological component, defined as data, a service, an application or a process in the model. For the conceptual models, we have developed a graphics-oriented finder and browser which allows to navigate in the information model. In addition, textual editors can be used to edit models using the language OML. For the customization we have integrated the ISA dialogue manager for GUI components and the MAESTRO database designer (MDD) for relational database management systems. This means we are using the generator components of ISA and MDD in combination with our own generator for distributed data, services, applications and processes. All the client parts of Ωmega are available for OS/2, the processes logic could also be distributed on several UNIX-platforms.

As indicated, the FAW software engineering paradigm focuses on the development of distributed software for heterogeneous computing environments, covering a large variety of computer systems and communication facilities together with multimedia components and telecommunication components as well. A neutralizing execution environment for this kind of communication infrastructure comes as part of the systems architecture. Taking into account that several standardization proposals are competing worldwide and large companies are seeking proprietary interoperability solutions, the FAW software engineering paradigm favours a neutralizing approach instead of a standardization approach to bring companies into the position to actively integrate heterogeneous environments for their applications. By using a knowledge based approach (runtime repository including knowledge about the communications infrastructure and the available data and methods) this execution environment - AMBAS - has been implemented at FAW. AMBAS provides a CORBA-compliant execution environment (Common Object Request Broker Architecture from the Object Management Group (c.f. OMG, 1991)) which has been shown in (Eck et al., 1994) and meets the CIMOSA-specification of an integrating infrastructure (Heimann et al., 1994). However, it provides additional functionalities, especially for intelligent request brokerage and integration of existing even non-CORBA-compliant networking environments, e.g. DCE, TCP/IP, SNA, NetBIOS and DECnet. The AMBAS execution environment also supports aspects of a knowledge-based API (Application Programming Interface), a distributed operating system, a distributed data-base management system, workflow control and monitoring functionalities.

The knowledge representation framework (Ωmega information model) is implemented as an AMBAS-object which can alternatively be made persistent in file systems, the OMS/PCMS-Repository (part of MAESTRO II) or ObjectStore (an object-oriented database management system). The architecture of AMBAS is depicted in the following Figure 9.

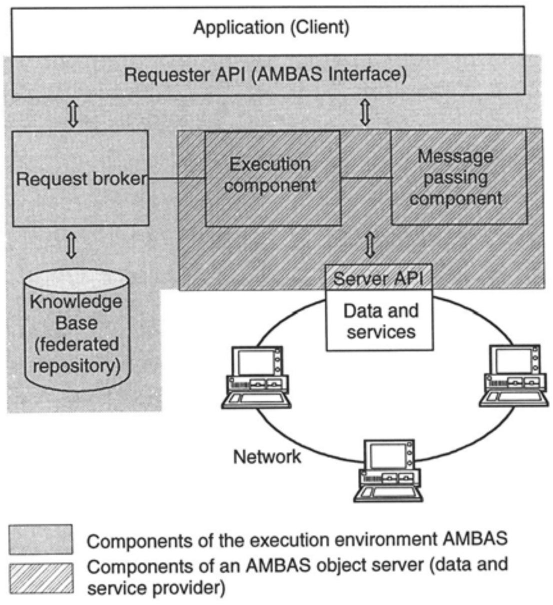
Application (Client)
Requester API (AMBAS Interface)
Request broker
Execution component
Message passing component
Knowledge Base (federated repository)
Server API
Data and services
Network
Components of the execution environment AMBAS
Components of an AMBAS object server (data and service provider)

Figure 9 AMBAS - Architecture overview.

AMBAS provides its application programming interface (Requester API) in a problem-oriented fashion. Applications can issue problems to the system instead of specifying the name and location of the function to be executed by specifying the output requested and the input that should be used. The intelligent request broker facility of AMBAS searches for matching methods and presents the most appropriate methods with respect to the problem description based on preference elicitation and other search strategies for selection. Current work done on Eigenmodel-based systems (c.f. Bartusch et al., 1989) applies the algorithmic concepts of Job-Shop Scheduling to this matching process (Möhring et al., 1994).

Based on the data stored in the information model AMBAS supports the development of new kinds of intelligent decision support systems in heterogeneous distributed computing environments by employing AMBAS′ intelligent method base facilities. AMBAS allows the distribution of data and methods and eases the integration of new components (data, services, applications and processes) into this framework. This builds the fundamental basis for cooperative development of software including decision support systems with the possibility to transfer new algorithms directly into industrial use.

Within the FAW software engineering paradigm a generator has been implemented to produce C++ for AMBAS-executables directly from OML-models as depicted in the following Figure 10.

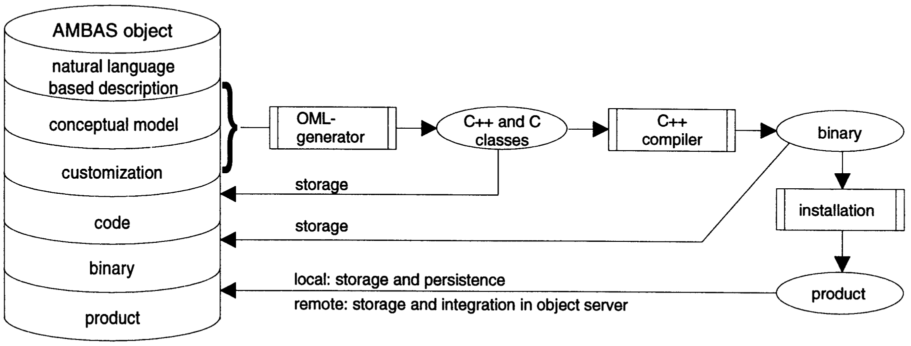

Figure 10 The Ωmega generator.

By using ObjectStore or a file system as the underlying technical framework for the knowledge administration, the generated C++-code can be made immediately persistent. Since data and methods (services, applications and processes) are implemented as objects the same way, AMBAS can be seen as a data and method base management system (extending the concepts of DBMS also to the administration of methods) with OML as its 4GL.

4 CONCLUSION

In this paper the main architectural components of the FAW Software engineering paradigm have been described. In addition, we gave an overview about the architecture of Ωmega and AMBAS, which implements this paradigm. This implementation of a neutralizing execution environment (AMBAS) and an engineering environment (Ωmega) has proven that an overall architecture for enterprise engineering, including process modelling and application development, can be built. Current work focuses on the enhancement of the prototypes. Several mechanisms (preference elicitation, graph-search et.) are added to the request-broker-component of AMBAS as well as monitoring-functions for intelligent load-balancing. Based on the complex modelling of the communication infrastructure knowledge, components of an autonomous and intelligent systems management especially for configuration, change & distribution are built to cover the needs to administrate large environments.

5 REFERENCES

Anderl, R. (1993) STEP-Grundlagen der Produktmodelltechnologie, in *Datenbanksysteme in Büro, Technik und Wissenschaft*, ed. W.A. Stucky.

Bartusch, M., Möhring, R. H., Radermacher, F. J. (1989) Design Aspects of an Advanced Model-Oriented DSS for Scheduling Problems in Civil Engineering; *Decision Support Systems*, Vol. 5, No. 4.

Eck, O., Heimann, M., Holocher, M. (1994) Abdeckung der OMG/CORBA-Spezifikation durch die FAW-Software Engineering Strategie, *FAW Technical Report*, FAW-TR-94015.

ESPRIT Consortium AMICE (Eds.) (1993) CIMOSA - Open System Architecture for CIM, *Research Reports ESPRIT*, Springer.

Grabowski, H. (1993) STEP als Integrationskern für die Produktdatengenerierung, *VDI-Zeitschrift* 135, Nr. 7.

Heimann, M., Holocher, M., Solte, D. (1994) AMBAS - A CIMOSA Compliant Execution Environment, Correspondence of the CIMOSA Integrating InfraStructure and the FAW Execution Environment AMBAS, *FAW Technical Report*, FAW-TR-94019.

Holocher, M., Michalski, R., Radermacher, F. J., Rapl, K., Solte, D. (1994) Gegenüberstellung von Konzepten der relationalen Modellierung der objektorientierten Modellierung und der vollen Modellierung, *FAW Technical Report*, FAW-TR-94002.

Holocher, M., Michalski, R., Solte, D., Vicuna, F. (1993) MIDA - An Open Systems Architecture for the Model-Oriented Integration of Data and Algorithms, *FAW Technical Report*, FAW-TR-93018.

Holocher, M., Solte, D. (1992) AMBAS - An Adaptive Method Base Shell; in *Enterprise Integration Modelling*, ed. C.J. Petrie, Jr., MIT Press.

Kaufmann, F. (1993) Erstellen von Modellen für Organisations- und DV-Lösungen, Entwurf und Spezifikation betrieblicher Objektsysteme mit der grafischen Entwurfssprache GRAPES, SNI-AG, Berlin, München.

Kopaczyk, A., Michalski, R., Rapl, K., Solte, D. (1993) Kommunikationsinfrastruktur 2000, *FAW Technical Report*, FAW-TR-93023.

Möhring, R. H., Müller, R., Radermacher, F. J. (1994) Advanced DSS for Scheduling: Software Engineering Aspects and the role of Eigenmodels; Proceedings *27th Annual Hawaii International Conference on System Sciences.*

Müller, R., Solte, D. (1994) *How to make OR-results available - a proposal for project scheduling*, will appear in special volume of annals of operations research.

OMG (1991) The common object request broker; architecture and specification, *OMG Document* No. 91.12.1.

Petrie, C.J. Jr. (Ed.) (1992) Enterprise Integration Modelling: Proceedings of the first international conference, MIT Press, Cambridge.

Radermacher, F. J., Solte, D. (1994) Die FAW-Software-Engineering Strategie für Multi-Client/Server-Umgebungen, Proceedings On-line '94.

Solte, D. (1987) Open Systems, Ein lernendes Verwaltungssystem für die rechnerunterstützte Methodenkonstruktion im Bereich des Operations Research, *VDI Reihe* 16, Nr. 38.

Solte, D., Heerklotz, K.-D. (1991) Knowledgebased Management of Distributed Ressources, *FAW Technical Report*, FAW-TR 91001.

Teufel, B. (1991) Organization of Programming Languages, Springer.

Verall, M. S. (1991) Unity Doesn't Imply Unification of Overcoming Heterogeneity Problems in Distributed Software Engineering Environments, *The Computer Journal*, Vol. 34, No. 6.

6 BIOGRAPHY

Dirk Solte received his doctoral degree after studying business engineering with focus on operations research and computer science at the University of Karlsruhe. Since 1988, he has been a senior scientist at FAW, heading the department "Communication Systems / Industrial Software Production" and co-heading the department "Enterprise Integration / Decision Support Systems". He is also responsible for the sophisticated technical infrastructure of FAW. This focuses his work in these domains to solutions in heterogeneous distributed environments. Dr. Solte has published several papers in these fields, directed a number of ambitious research and software development projects and consulted industry.